AOSI

Intro:
Distributed Operating-Systems

roadmap:

- Characteristics of distributed systems
- Models of communication and sharing
- Distributed Shared Memory (DSM)
- Remote Procedure Call (RPC) and Remote Method Invocation (RMI)
- Distributed File Systems (NFS, AFS)
- Security and Protection
- Order and time in distributed systems

Multi-Processor Systems

Bus-based Multi-Processor with single central memory.

Realization: Hardware.
Problems: Cache coherence and memory consistency.

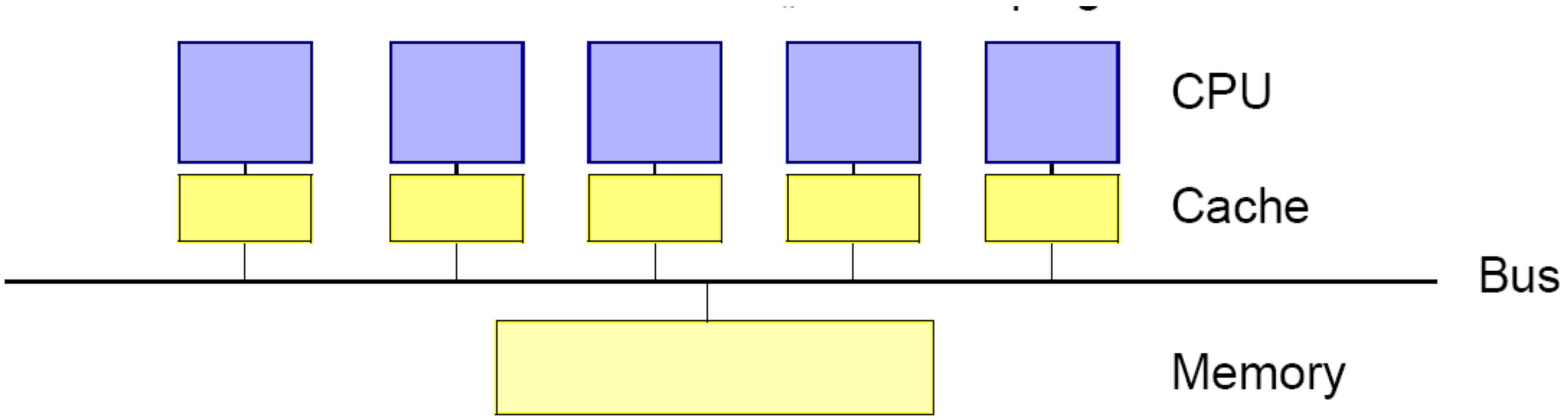
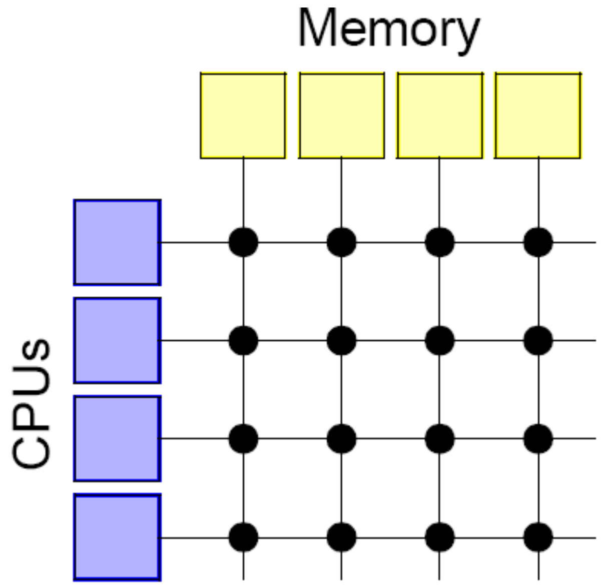


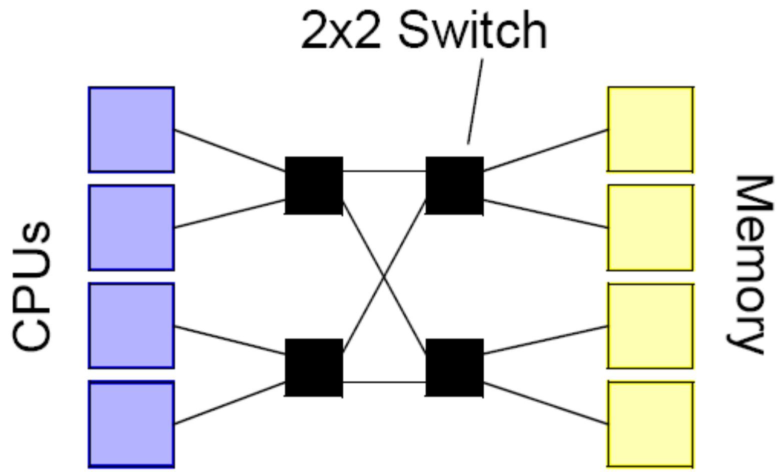
Abb.: F. Hauck: Modernen Konzepte verteilter Systeme (VL U.Ulm, 2002/02)

Multi-Processor Systems

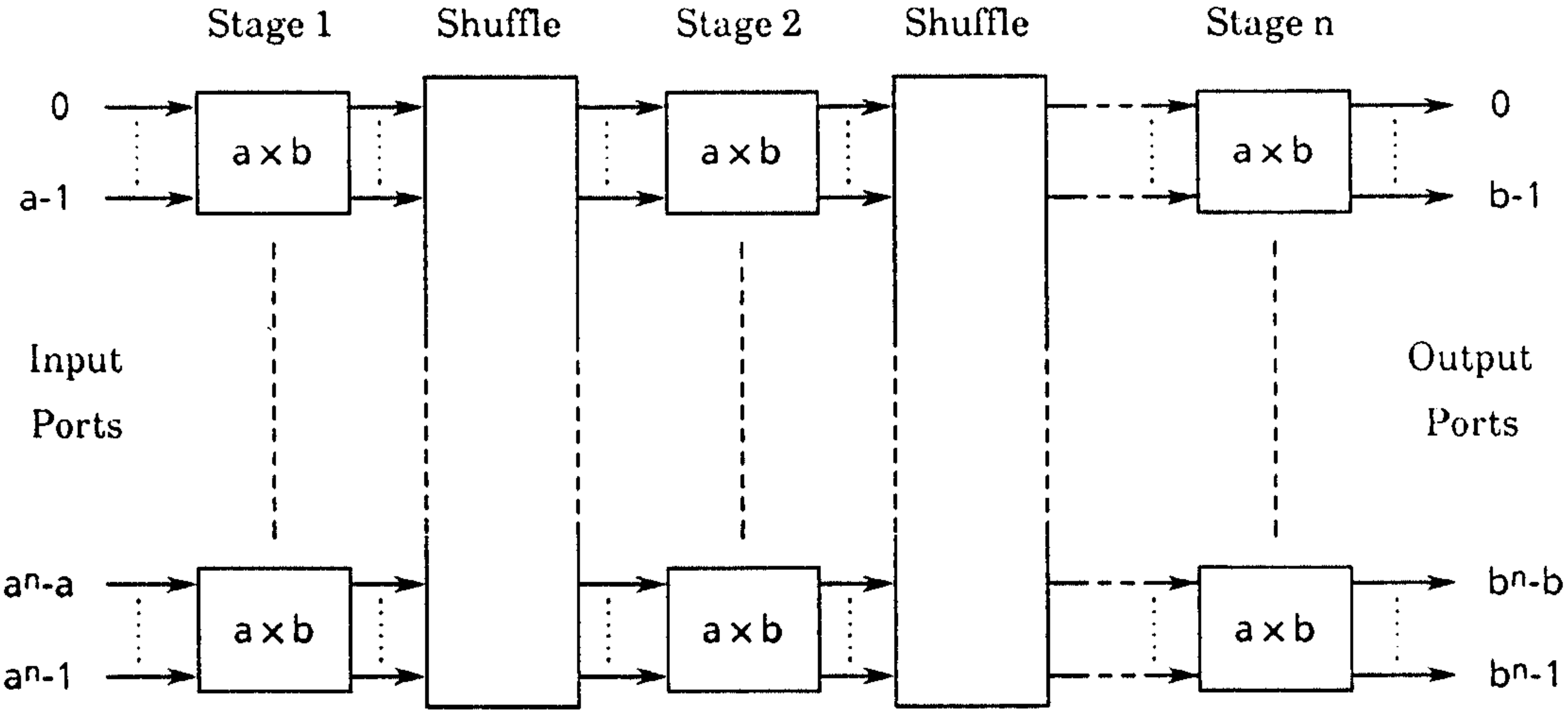
Connection-based Multi-Processor with multiple memories.

Realization: Special switching network hardware (Omega networks, Banyan trees,..)
Problems: Complexity of the switching network.

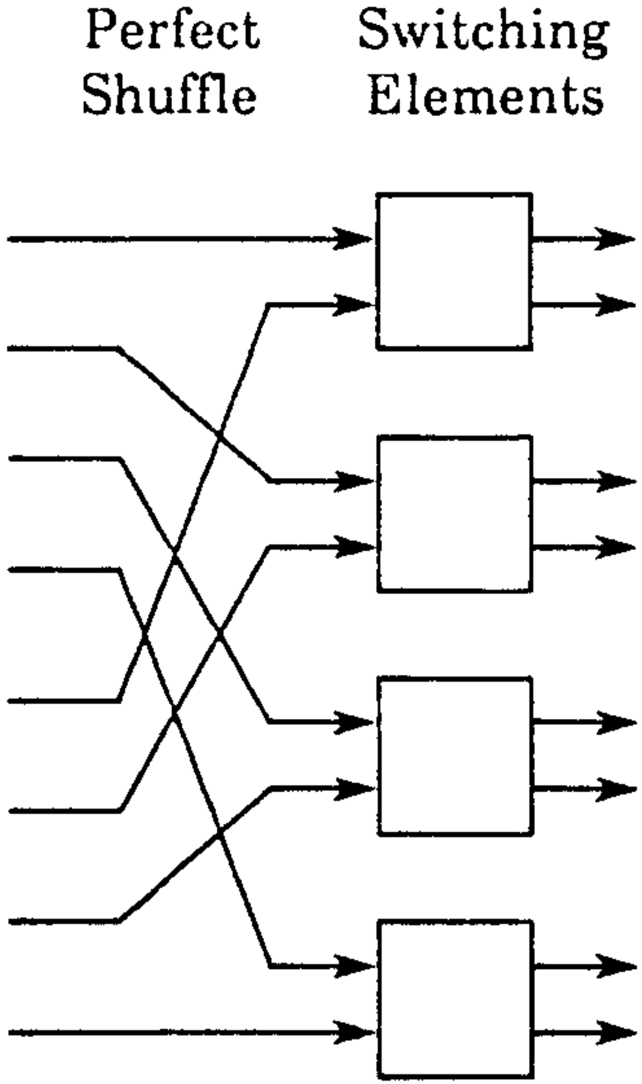




General Form of a Delta Network



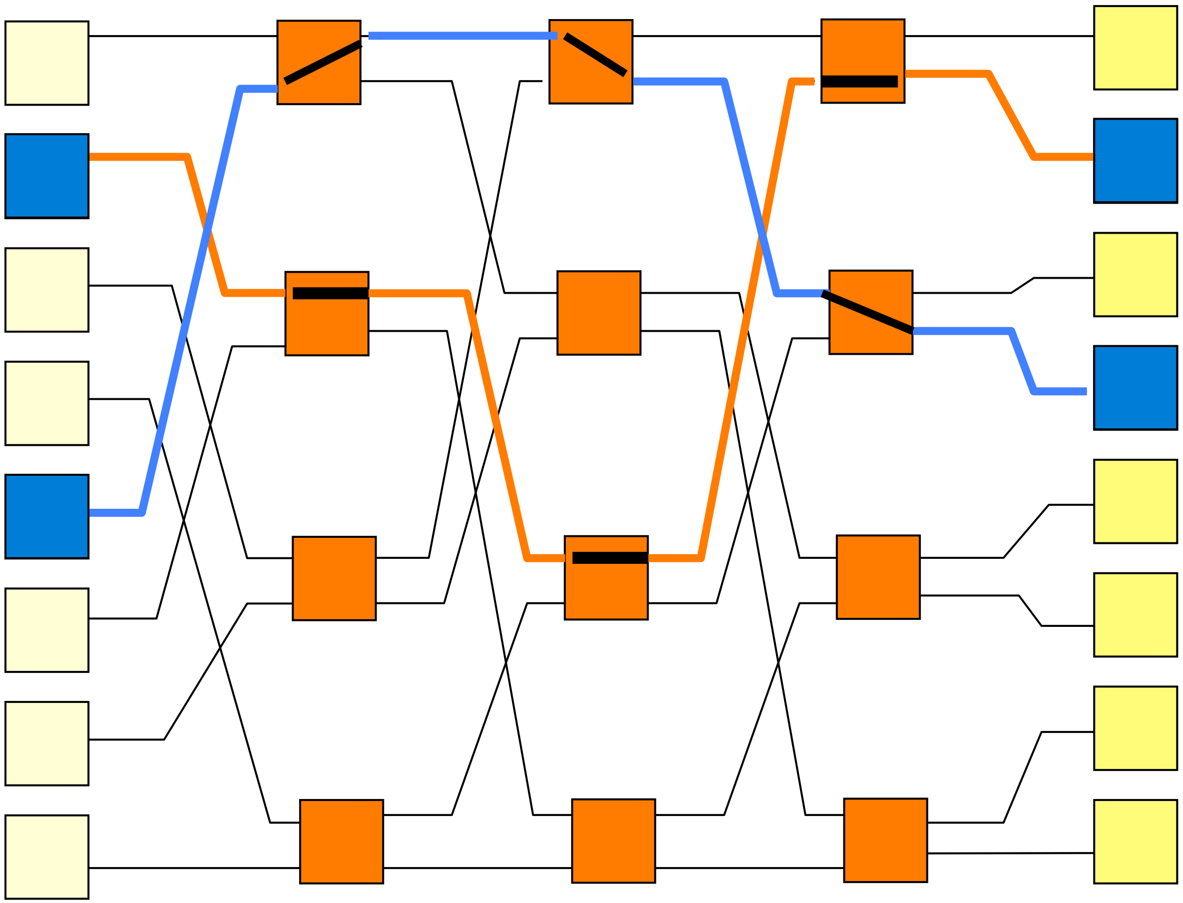
The shuffle stage



An Omega switching network

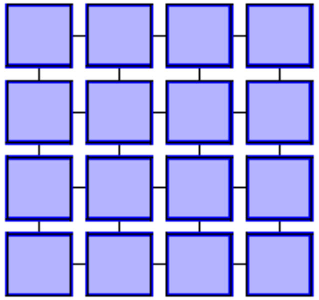
$\log_2 n$ stages

$2^k = N$ inputs

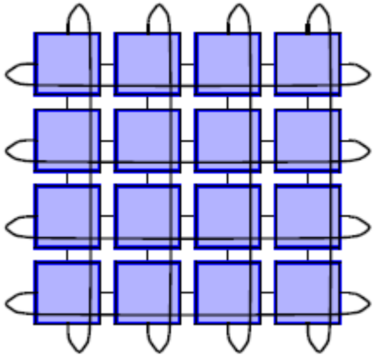


Multi-Processor Topologies

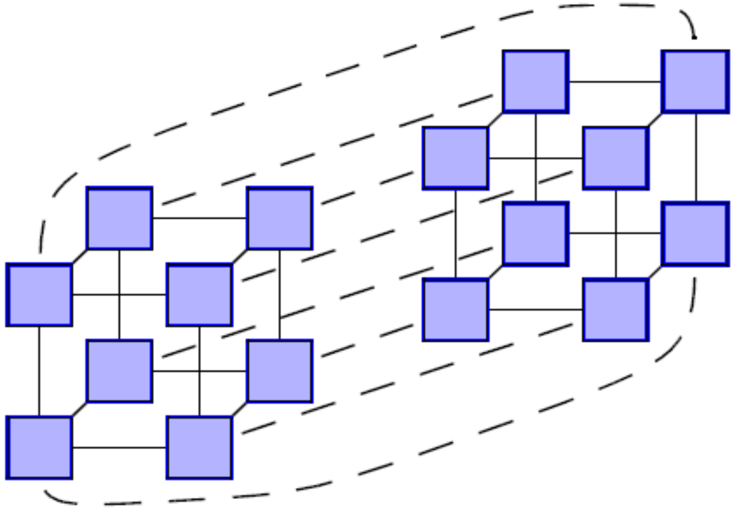
Grid



Torus



Hypercube



	max. distance
Grid	6
Torus	3
Hyperc.	3

Types of Multi-Processor Systems

	data	control	
shared memory multiproc.	c	c	tight coordination of multiple execution engines
computer cluster	d	c	central coordination of proc/mem pairs working on distributed data
distributed system	d	d	no central component.

What is a distributed system?

Leslie Lamport: You know you have one when the crash of a computer you have never heard of stops you from getting any work done.

Andrew Tanenbaum: A distributed system is composed from multiple autonomous computers which appear as a single computer for a user.

George Coulouris: A distributed system is composed from multiple autonomous computers which coodinate actions by exchanging messages.

What is a distributed system?

Essential properties:

- multiple computers (local CPU-/memory-/network-/I-O-components)
- computers are autonomous, i.e. they have an independent local control
- computers are connected by a network and basically communicate by exchanging messages
- there is no special central control and coordination facility

Distributed Data + Distributed Control

What is a distributed system?

Essential properties:

- **Concurrency of computations**
- **No global time (approximations possible)**
- **Components fail independently**

Why a distributed system?

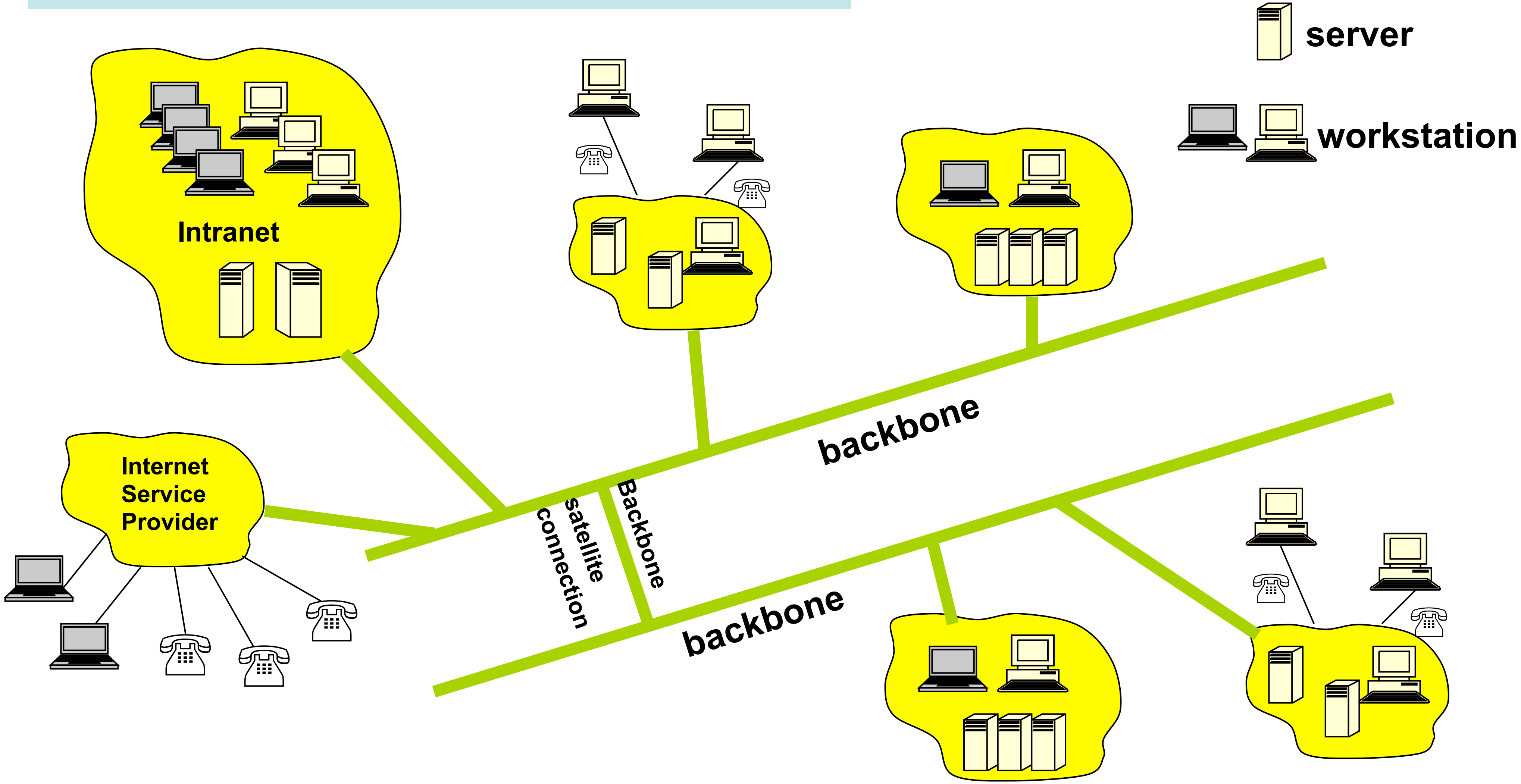
Issues in distributed system:

- Performance
- Sharing of resources
- Independence of failure and no single point of failure
- Distributed nature of application
- Distributed data
- Extensibility and Scalability

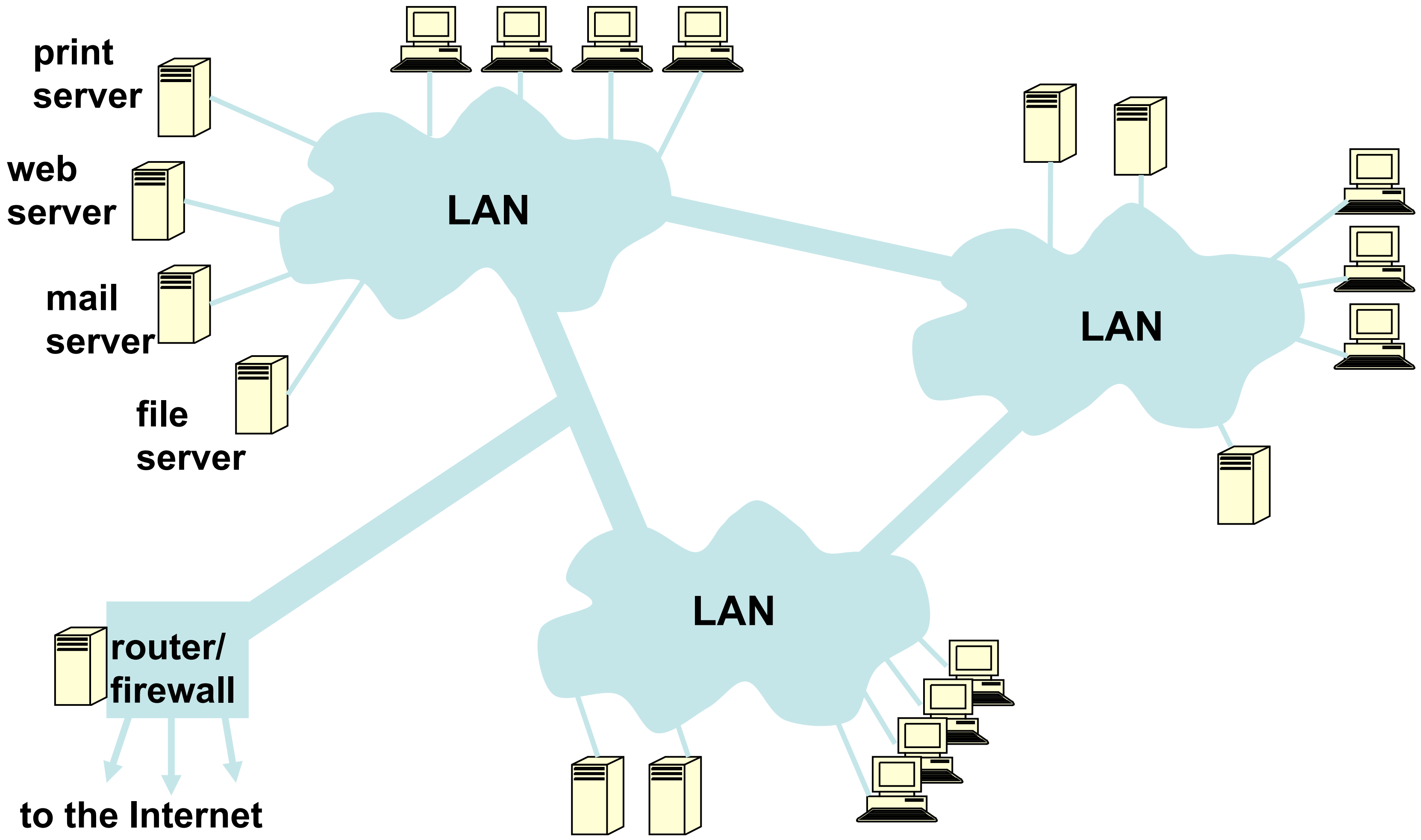
Examples

- The Internet
- An Intranet
- Distributed Control Systems
- Ubiquitous and mobile computing environments

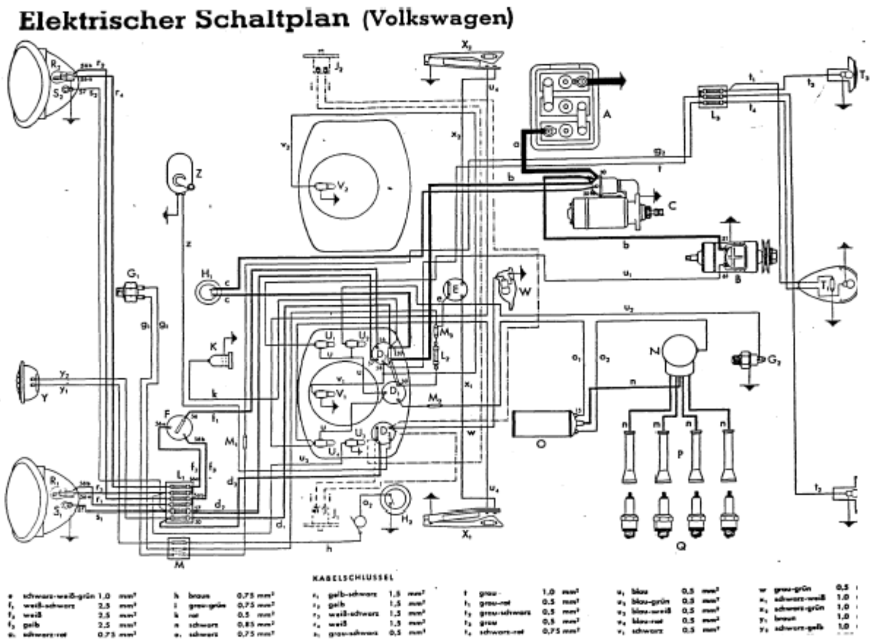
Example: Internet



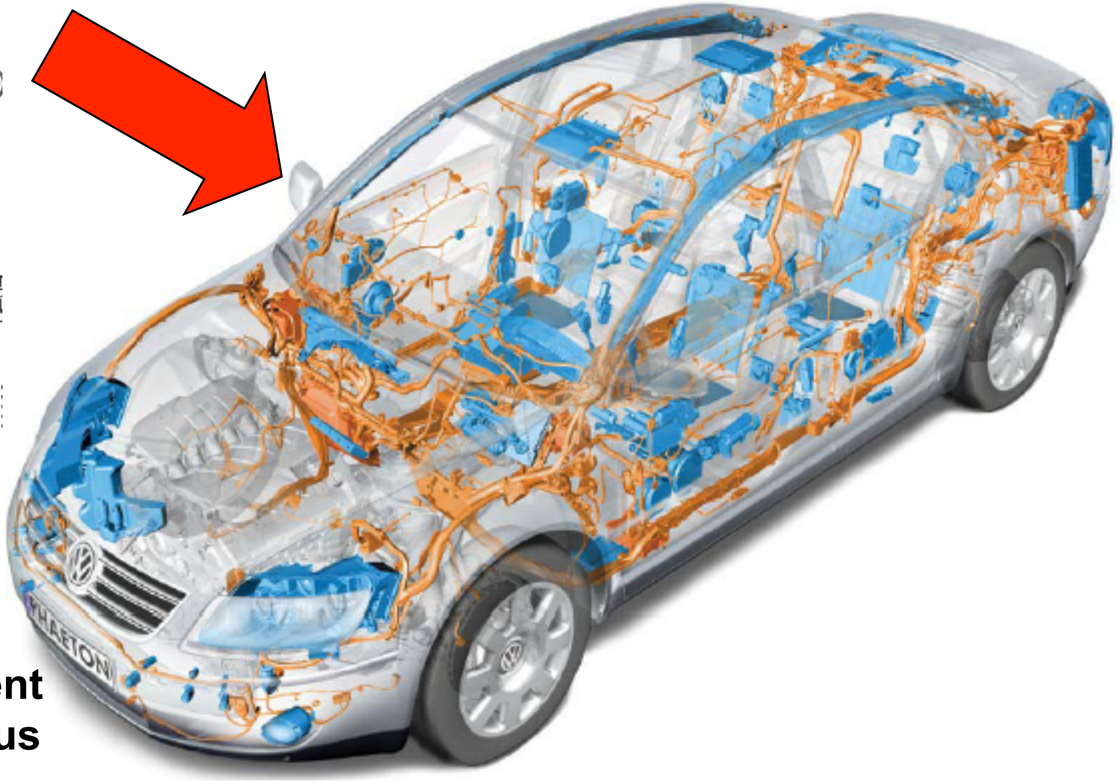
Example: Intranet



Example: Control Networks

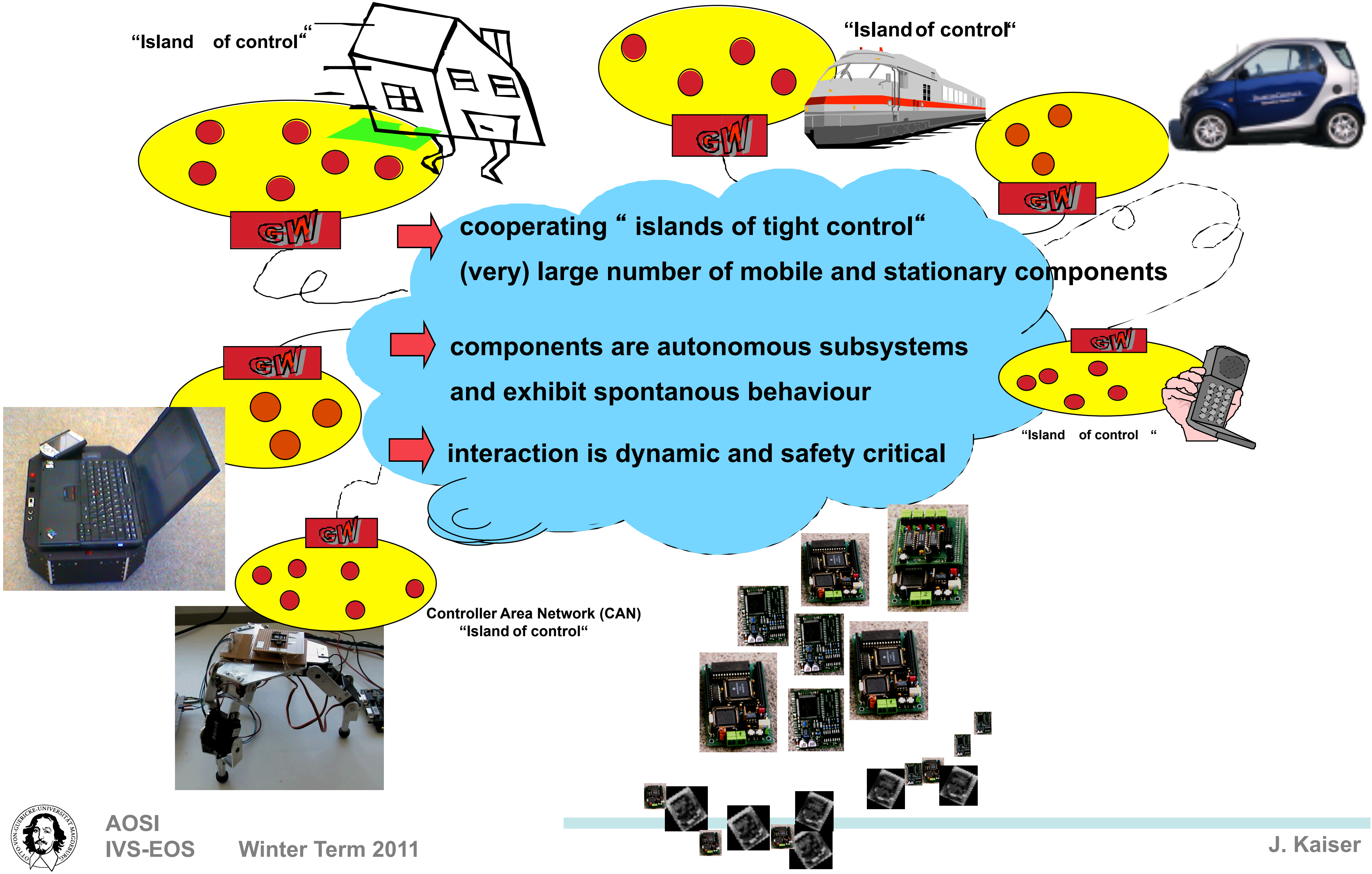


drastically increasing complexity



- **11.136 electrical parts**
- **61 ECUs**
- **Optical bus for information and entertainment**
- **Sub networks based on proprietary serial bus**
- **35 ECUs connected to 3 CAN-Busses**
- **2500 signals in 250 CAN messges**

Example: A networked physical world



"Island of control"

"Island of control"

"Island of control"

Controller Area Network (CAN)
"Island of control"

cooperating " islands of tight control"

(very) large number of mobile and stationary components

components are autonomous subsystems

and exhibit spontanous behaviour

interaction is dynamic and safety critical

Problems and desirable properties

- **general problems:** **concurrency, delays, faults**
- **more problems:** **heterogeneity, openess, scalability**
- **desirable properties:**

A distributed system should be programmable like a local, centralized computer (→ see Tanenbaum).

???

- **Support to deal with the above problems in an application specific way on an adequate level of abstraction.→ Find a better definition!**

Transparencies:

- Access transparency
- Location transparency
- Concurrency transparency
- Migration transparency
- Relocation transparency
- Replication transparency
- Fault transparency
- Persistency transparancy

?

Qos transparency

Types of distributed operating systems

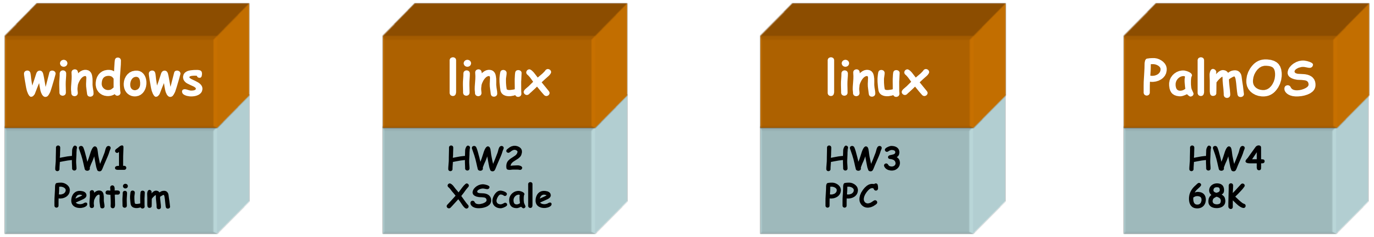
Network operating systems: basic support for communication between homogeneous local OS, individual computing nodes are visible
Examples: Windows NT, UNIX, Linux,
distributed file systems (NFS)

Distributed operating systems: transparent IPC mechanism, no difference between local and remote interaction, unified name space, integrated file system, unified user admin and protection/security mechanisms.
Examples: Amoeba, Emerald, Chorus, Clouds

Middleware: builds on top of heterogeneous local OS, provides unified programming model, communication and cooperation mechanisms, maintains autonomy of local nodes but supports transparent access to shared resources.
Examples: CORBA, Java RMI, .NET, DCE

Distributed system architecture

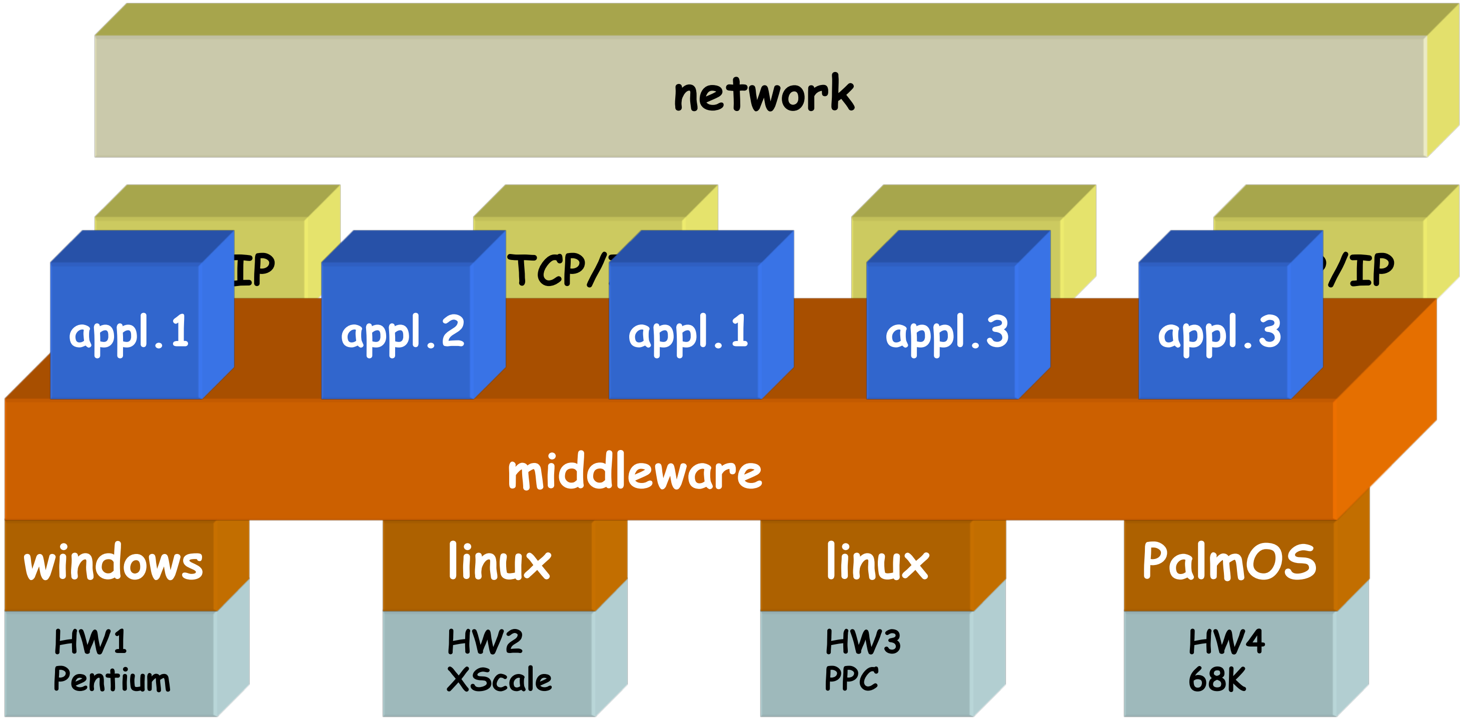
abstracting from HW



Distributed system architecture

abstracting from local OS

abstracting from HW



Types of middleware

Document-based middleware:
model: distributed data

Documents which contain (hyper-) links to other documents.
Examples: World-Wide-Web

File-based middleware:
model: distributed data

Transparent access to remote files.
Examples: Andrew File System, NFS

Object-based middleware:
model: distrib. functions

Transparent invocation of remote objects.
Examples: CORBA, DCOM(windows only)

Service-based middleware:
model: distrib. functions

Discovery and use of remote services.
Examples: Jini, JXTA, UPnP

Coordination-based middleware:
model: distrib. functions

Coordination through a shared information space.
Examples: Linda, Java Spaces, Lime